Seesaw

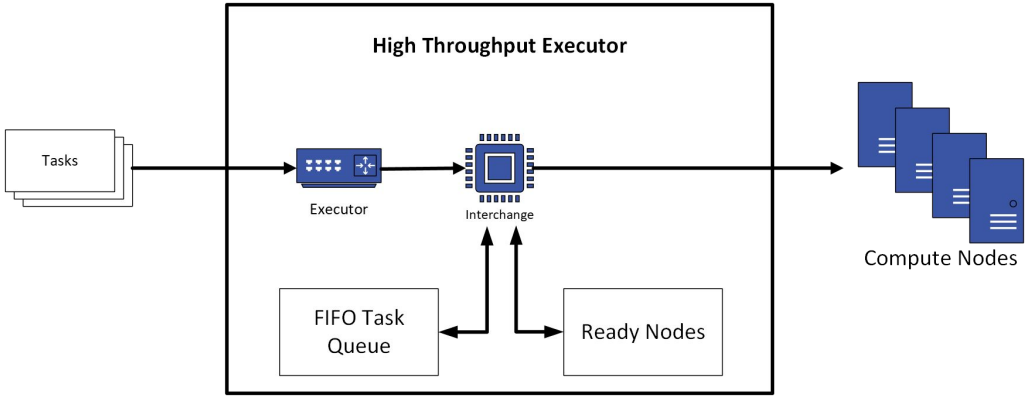
Elastic Scaling for Task-Based Distributed Programs

Matthew Chung

UC RIVERSIDE

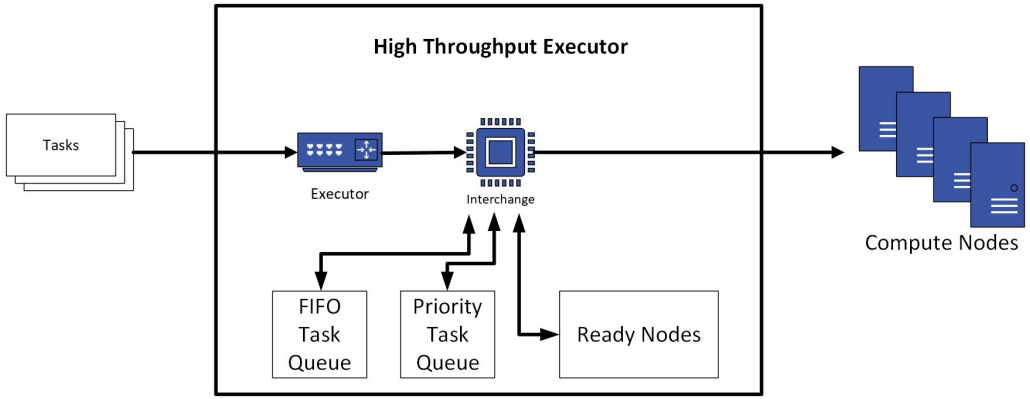
Current Implementation

- Random selection from list of available nodes
- First in, first out queue for tasks
- Good workload distribution
- Poor elastic resource provisioning



Updated Implementation

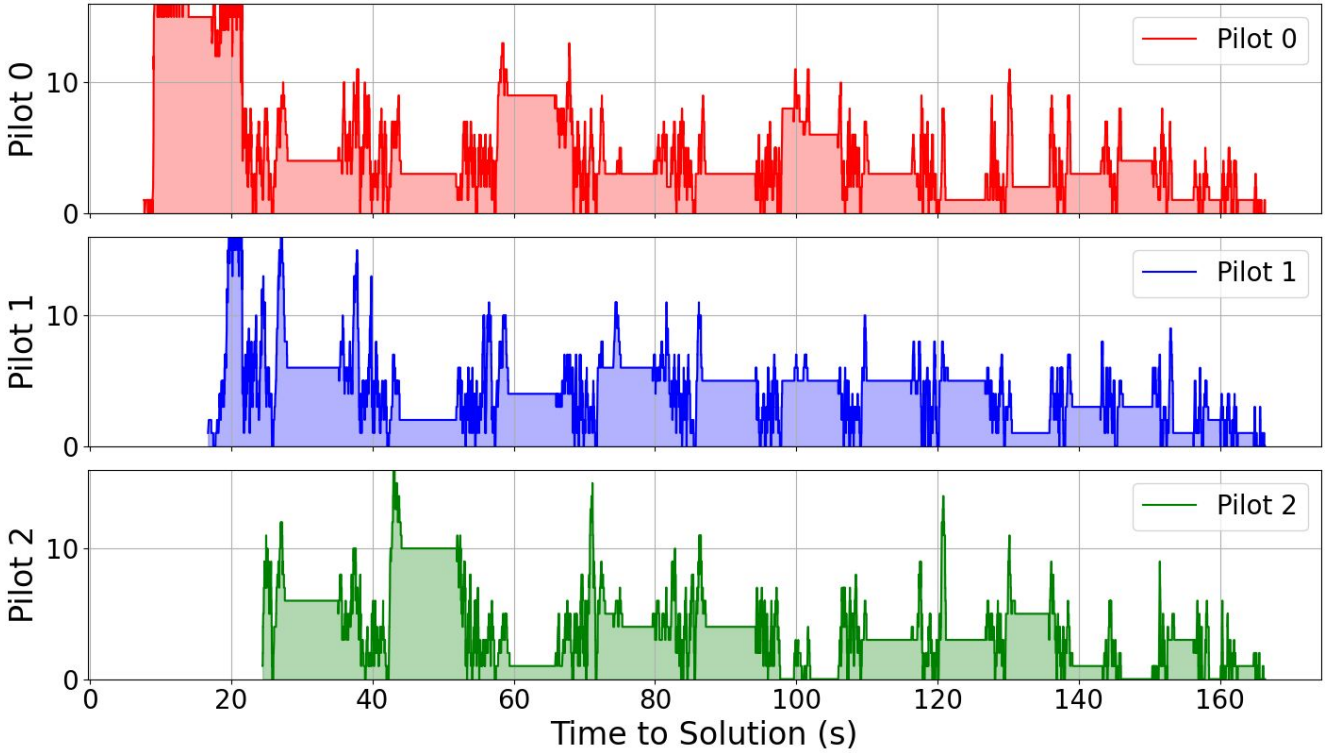
- Two queues for tasks
  - Priority runtime-based queue
  - Original FIFO queue
- Sorted list of ready nodes
- Improved scaling behavior
- Higher resource utilization



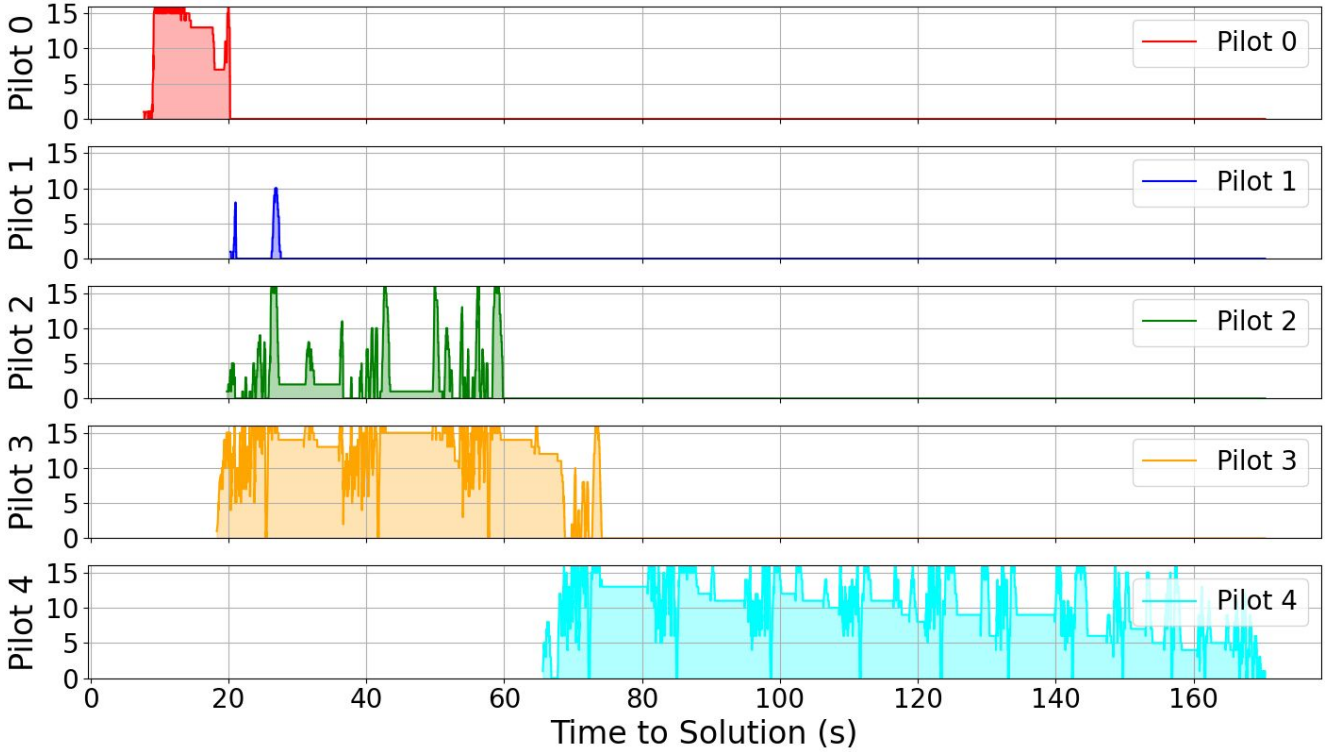
Testing Methodology

- **Tested two different types of workloads:** Cholesky Factorization and Synthetic
- **Cholesky Factorization:**
  - Representation of a dataflow based workflow
  - Varying influx of tasks over time
  - Run through TaPS
- **Synthetic:**
  - Representation of a bag-of-tasks workload
  - 3,000 sleep tasks with runtimes between 0 and 140 seconds
  - Log-normal right skew distribution

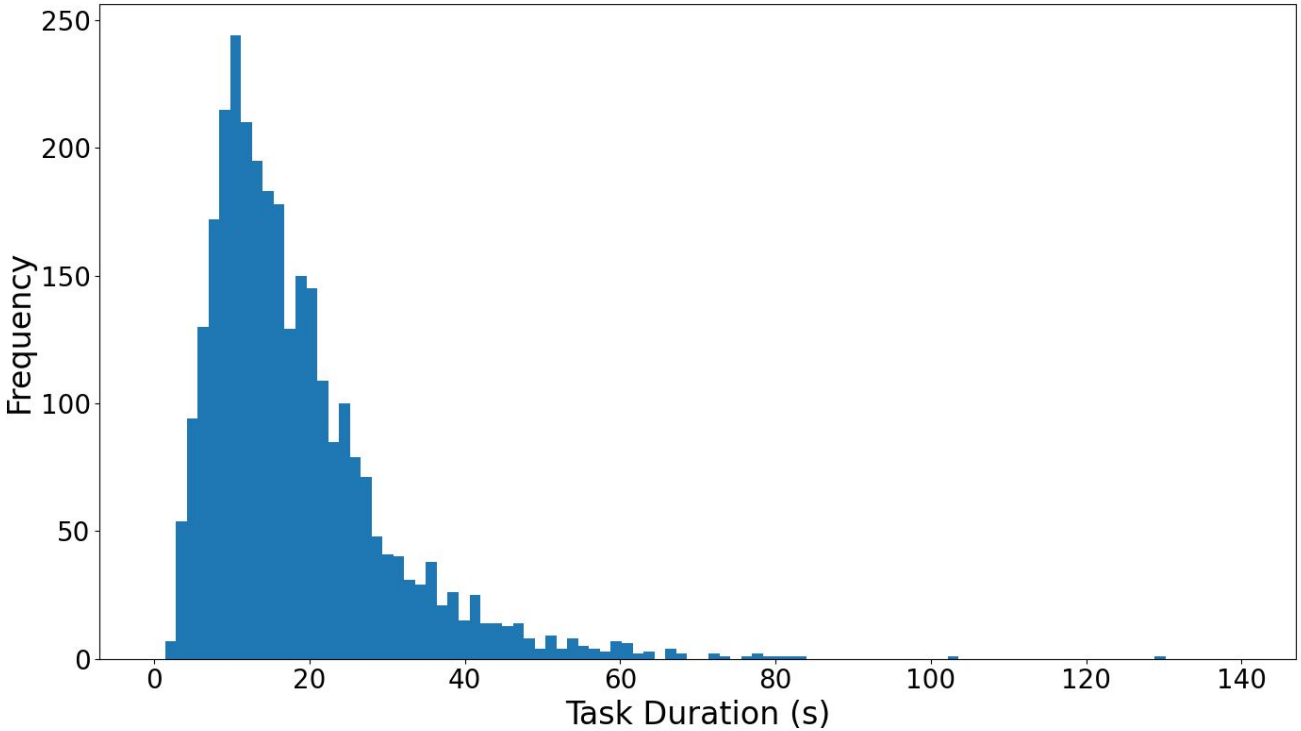
Cholesky Results - Random



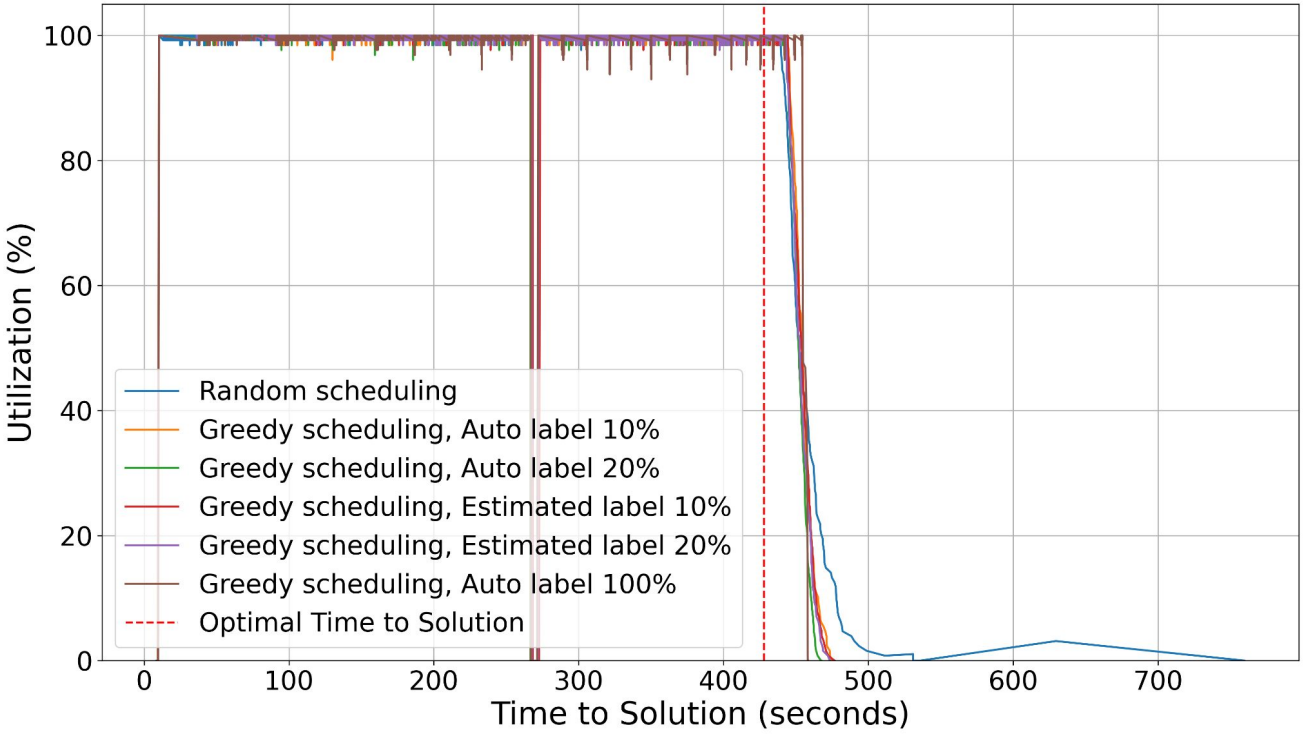
Cholesky Results - Seesaw



Synthetic Distribution



Synthetic Results



Analysis

- **Cholesky Factorization:**
  - Significant improvement in ability to scale down
  - Similar time-to-solution
  - Reduction in compute resource usage, increase in utilization
- **Synthetic:**
  - Node sorting does not have much effect
  - Task sorting greatly improves time-to-solution
  - Utilization remains high in both methods

Next Steps

- Modular task labeling system
- Testing other workloads
- Guide/documentation update for new features

Questions?

Thank you for listening!

UC RIVERSIDE